VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce očního bělma v obrazech

Sclera Detection Based on Image Analysis

2017 Aleš Joska

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student:	**Aleš Joska**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	2612R025 Informatika a výpočetní technika
Téma:	Detekce očního bělma v obrazech Sclera Detection Based on Image Analysis
Jazyk vypracování:	čeština

Zásady pro vypracování:

Detekce očního bělma v obrazech může sloužit jako doplňková funkce pro detekování otevřených nebo zavřených očí nebo například pro zpřesnění lokalizace duhovky a zornice. Tato detekce může být také dále využita jako součást detektoru, který analyzuje a vyhodnocuje únavu řidiče v automobilu.

1. Seznamte se se základními pojmy v oblasti analýzy a detekce lidských očí v obrazech, zejména se zaměřte na obrazové příznaky a segmentační metody.
2. S pomocí knihovny OpenCV jedno vhodné řešení implementujte.
3. Experimentálně ověřte funkčnost, přesnost a rychlost navrženého detektoru.
4. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

[1]Erdogmus, N., and Dugelay, J.: An Efficient Iris and Eye Corners Extraction Method. Structural, Syntactic, and Statistical Pattern Recognition, pp. 549–558 (2010)
[2]Radu, P., Ferryman, J., Wild, P.: A robust sclera segmentation algorithm, in Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on, pp.1-6 (2015)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek, Ph.D.**

Datum zadání: 01.09.2016
Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017

Aleš Jaška

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 ..... Aleš Jaška .....

Rád bych na tomto místě chtěl poděkovat vedoucímu mé bakalářské práce Ing. Radovanu Fusekovi, Ph.D. za odbornou pomoc, trpělivost a cenné rady.

Abstrakt

Tato bakalářská práce se zabývá problematikou detekce očního bělma v obrazech. Hlavním cílem je popsat využívané metody detekce v reálném prostředí, jeden vhodný algoritmus realizovat a ověřit na volně přístupných vzorech. Pro realizaci aplikace je použit programovací jazyk Python s knihovnou OpenCV a s klasifikátory Haar Cascade, které jsou nutné pro získání pozic očí a lidské tváře. Popsány jsou metody na bázi prahování obrazu a detektorů kontur, dále algoritmy pro vyhledávání podobně zbarvených bodů a predikce pomocí význačných bodů. Součástí práce je i porovnání úspěšnosti jednotlivých detekcí.

Klíčová slova: OpenCV, zpracování obrazu, detekce bělma, prahování, význačné body

Abstract

This thesis deals with problematics of sclera detection inside images. The main aim of this work is describe of available sclera detection methods, realize one useful algorithm and verify it on on accessible samples. For implementation of application is utilized programming language Python with OpenCV library and Haar Cascade Classifiers, which are necessary for eyes and face detection. In thesis are described methods on thresholding bases and contour detection, algorithm for finding similar colored pixels and facial landmarks predictor. Part of this thesis is also comparison of success of individual detection

Key Words: OpenCV, image processing, sclera detecion, thresholding, facial landmarks

Obsah

Seznam použitých zkratek a symbolů

AdaBoost	–	Adaptive Boosting algoritmus
BGR	–	Blue-Green-Red barevný model používaný v OpenCV
BSD	–	Berkeley Software Distribution licence
CGDS	–	Columbia Gaze Data Set
CIE	–	Barevný model spravovaný mezinárodní komisí pro osvětlování
HF	–	Haar Features, neboli Haarovy příznaky
HOG	–	Histogram of Oriented Gradients, neboli histogram orientovaných gradientů
HT	–	Houghova transformace
HSV	–	Hue-Saturation-Value model
LBP	–	Local Binary Pattern, neboli lokální binární vzor
MSER	–	Maximálně stabilní extrémy regionu (Maximally stable extremal regions)
OpenCV	–	Open-Source Computer Vision
OT	–	Otsu transformace (nebo binarizace, nebo také prahování)
RGB	–	Red-Green-Blue barevný model

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době můžeme zpozorovat aktivnější účast počítačových technologií, které zpracovávají lidské části těl v obrazech. Například při monitorování obličeje postavy máme možnost vyčíst mnoho informací o sledované osobě. Můžeme třeba zjistit, zda má člověk otevřené oči, detekovat mrkání, směr který osoba sleduje, nebo složitějším pozorováním vyčíst výrazy a emoce postavy.

Jednou z možností, jak základní informace určit je detekce očního bělma. Například pokud se nám podaří vypočítat obsah bělma na stranách oka, můžeme pomocí matematických operací porovnat tyto veličiny a zjistit, která strana od duhovky má nejmenší výplň. Právě tato část definuje směr, který osoba sleduje. Obsah bělma můžeme například v intervalech dále porovnávat. Pokud veličina celkového počtu pixelů skléry bude velmi nízká, až nulová, tak má osoba zavřené oči.

Detekci očního bělma můžeme využít v mnoha oborech. Například v dopravních aplikacích existuje možnost pomocí mrkání, či přimhouřených očí ověřovat, zda řidič nepřešel do stavu mikrospánku. Tím lze potencionálně spustit poplašný systém, upozorňující na nutný odpočinek a zabránit tak dopravní nehodě. V lékařských aplikacích, zabývajících se sledováním pacientů je možné nalézt aplikace s algoritmy určené pro vyhledání červených žilek v bělmu. V neposledním případě jsou detektory na bázi sledování obsažené v bezpečnostních aplikacích, kde pomocí citlivých změn v mimice jsou kontrolovány reakce osoby.

Tato bakalářská práce je právě na způsoby detekce očního bělma zaměřená. Úvodem je stručně představená knihovna OpenCV, která se zabývá počítačovým viděním. Algoritmy této knihovny jsou využité v realizaci praktické části. Práce také obsahuje seznámení s jednotlivými metodami zpracování obrazu, které jsou ve většině detektorů implementované. Dále jsou popsány používané metody detekce bělma, které využívají prahování barev, detektory kontur a automatické prahování s využitím algoritmů pro rozpínání určené buňky do prostorových maxim. Také je popsaná metoda využívající význačné body v obličeji.

Závěrem je zhodnocena většina realizovaných metodik a příklady na veřejně přístupné databázi lidských tváří CGDS.

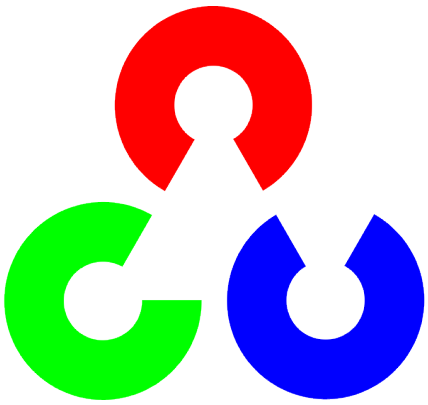
2 Detekce objektů v obrazech

Existuje mnoho systémů, které dokážou zpracovávat obraz. Jeden z volných a nejrozšířenějších softwarů je OpenCV. Tato knihovna poskytuje metody, díky kterým můžeme vyhledávat objekty v reálném čase.

Pro nadcházející detekci bělma je nutné, abychom nalezli obličej a oči v něm obsažené. Je samozřejmě možné detekovat bělmo i bez znalostí pozic těchto objektů, ale tím se při segmentaci dopracujeme k vysokému počtu negativních výsledků. Tyto výsledky mohou být například reprezentovány rušivými vlivy, kde vymezené světlé oblasti v lidské tváři se budou zaznamenávat jako bělmo. Podobně světlé (resp. světlošedé) zbarvení mohou mít například zuby.

2.1 Knihovna OpenCV

OpenCV (Open Source Computer Vision) Library je svobodná knihovna pod BSD licencí, která je určená pro práci s počítačovým viděním a pro zpracování obrazu. První verze vyšla v roce 2006, pod taktovkou firmy Intel. Knihovna je realizována v prostředí C a C++, ale díky své multiplatformnosti je podporováno využití i v dalších programovacích jazycích, jako například v Pythonu a v Javě, nebo v aplikaci MATLAB. V těchto prostředích má OpenCV nyní více než 300 metod zabývajících se analýzou obrazu. Díky svému stálému vývoji roste počet softwarů s touto knihovnou a o jeho budoucnost není aktuálně nouze. [1]



Obrázek 1: Logo OpenCV

2.2 Detekce tváře a očí pro následnou analýzu bělma

Tato kapitola je věnovaná technikám, které se využívají pro detekci předem určených objektů v obrazech. Převážná většina detektorů totiž vyhledává objekty pomocí klasifikátorů, které jsou sestrojené z příznaků a natrénovány na mohutné množině pozitivních i negativních snímků. Výhodou práce s příznaky oproti bodům (resp. pixelům) je urychlení procesu detekce.

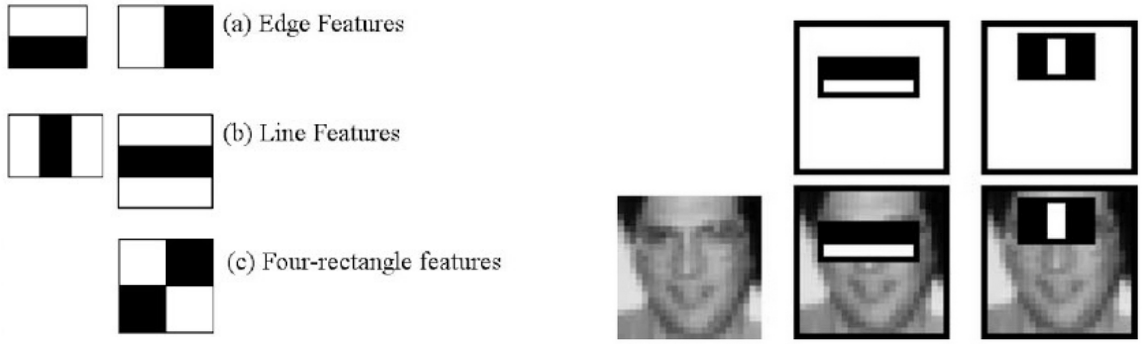
Nejčastěji se využívají následující typy příznaků:

- Haarovy příznaky (HF)
- Lokální binární vzor (LBP)
- Histogram orientovaných gradientů (HOG)

I přes to, že klasifikátor sestrojený z lokálních binárních vzorů je v porovnání s ostatními detektory výpočetně nejjednodušší, se častěji využívají Haarovy kaskádové klasifikátory, sestrojené z HF, které jsou efektivnější a odolnější vůči šumu, protože nepracují s hodnotami samotných pixelů, ale s obdélníkovými oblastmi. Také jsou součástí knihovny OpenCV a obsahují již serializované data, určené pro detekci očí a tváří. [2]

Rovněž je možné využít i HOG, ale ten se častěji vyskytuje v detektorech celých postav, nebo v pokročilých analyzátorech, využívající význačné body v obličejích. HOG má rovněž mírný problém s obrazy horší kvality. Proto praktická část práce využívá Haarovy kaskádové klasifikátory.

Jako první využili Haarovy příznaky pro detekci objektů v obraze Paul Viola a Michael Jones v roce 2001 pro detekci lidských tváří. [3]



(a) Typy použitelných Haarových příznaků (b) Výpočet jasových rozdílů

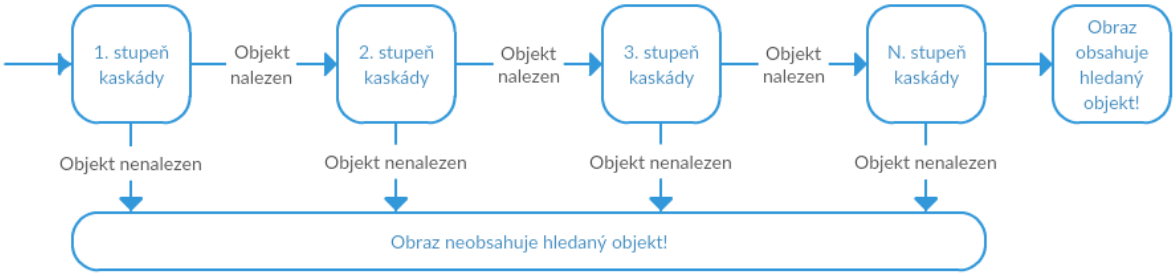
Obrázek 2: Operace s Haarovými příznaky[1]

Příklady Haarových příznaku jsou na snímku 2a. Jednoduše řečeno, jedná se o součty pixelů ležící v bílé části plochy odečtené od součtu pixelů z černé plochy. Těchto příznaků může být v 24x24 okně až 16000. Proto bylo sestrojen tzv. integrální obraz, který tuto sumarizaci zjednodušuje. [3]

Poté je sestrojen kaskádový klasifikátor. Ten je ze začátku tvořen slabými klasifikátory, které jsou utvářeny malým počtem příznaků. Ty jsou poté sloučeny do silného klasifikátoru. Pomocí algoritmu AdaBoost je klasifikátor natrénován. Každé z oblastí je přidělena váha, která se po každém uspořádání nesprávně klasifikovaným snímkům zvětšuje.

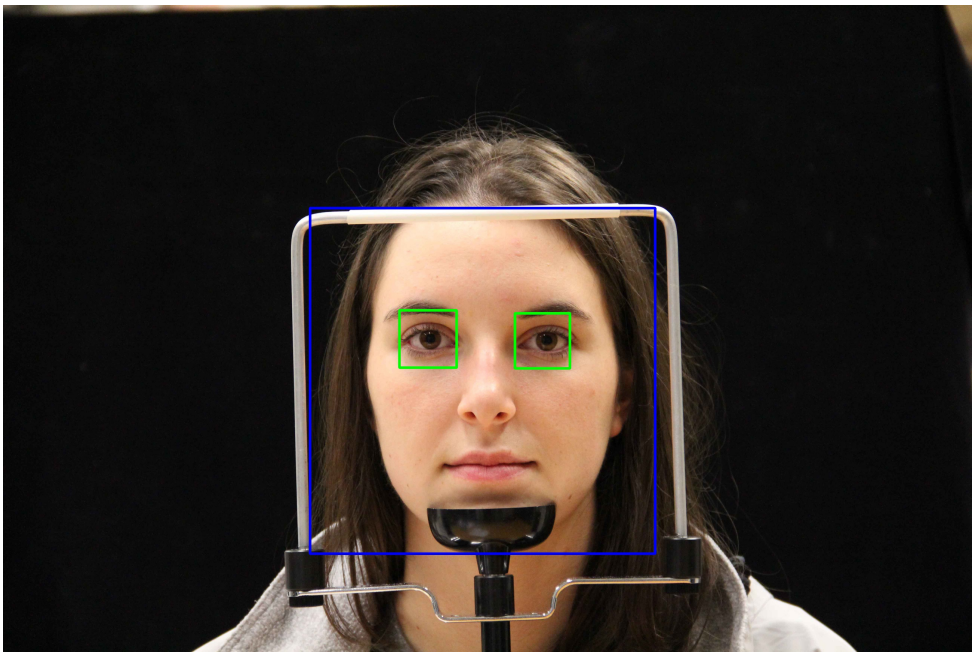
[1]Převzato ze článku *Face Detection using Haar Cascades* umístěného v dokumentaci OpenCV

Nejprve jsou odstraněny všechny negativní oblasti, jež neobsahují hledaný objekt. Výsledný zbytek oblastí pokračuje k vytvoření stupně klasifikátoru. Diagram postupu je na snímku č. 3.



Obrázek 3: Postup sestrojení kaskádového klasifikátoru

Po aplikaci klasifikátorů určených pro detekci tváře a očí, bychom v ideálním případě získali následující výstup:



Obrázek 4: Vyhledání vrstvy obličeje a očí v CGDS pomocí Haarových kask. klasifikátorů

3 Využívané modely při detekci objektů v barevném prostředí

Barvy jsou v počítačových systémech definovány pomocí barevných modelů. Ty určují, jakým dalším způsobem se tvoří výsledné barvy každého bodu.

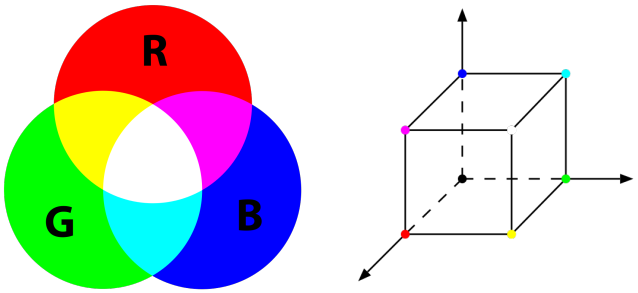
Nejčastěji využíváme RGB model, ale detetektory využívající knihovnu OpenCV jen zřídka kdy pracují se segmentací v RGB, kvůli reprezentování barev. Proto je v detektorech často implementován převod do jiného prostředí, nejčastěji však do HSV. V něm se totiž obraz matematicky lépe vnímá, jak si představíme v kapitole: *Barevné modely HSV a HLS* (č. 3.2)

3.1 Barevný model RGB

RGB je v aktuální době nejrozšířenější barevný model, který využívá kombinace červené, zelené a modré barvy. Je založen na teorii Thomase Younga a Hermanna von Helmholtze z 19. století, kteří vyzkoumali, že lidské oko je citlivé právě na tyto tři barvy. [4]

Jedná se o aditivní model, což znamená, že můžeme sestavit téměř jakoukoliv barvu smícháním všech tří složek. Promíchání všech složek v nejvyšší intenzitě vytváří bílou barvu, zatímco složky v nejnižší intenzitě vytváří černou barvu. Model RGB je možné geometricky představit jako krychli, ve které každá z kolmých hran udává škálu mohutností barevných složek (obr. 5b).

V počítačových systémech se RGB prostor udává pomocí 24 bitů, kde první osmice je určená pro červenou barvu, následující osmice pro zelenou barvu a poslední osmice pro modrou barvu.



(a) Příklad RGB aditiv

(b) Geometrické znázornění RGB

Obrázek 5: Znázornění Red-Green-Blue modelu[1]

RGB ovšem není základním prostorem v knihovně OpenCV. Tím je totiž model BGR, který pracuje se stejnými složkami, ale s jejich rozdílným uspořádáním, kde první osmice bitů modelu byla určena pro modrou barvu místo typické červené. Příčinou využívání BGR modelu jsou historické důvody. BGR byl populární u poskytovatelů a výrobců softwaru pro fotoaparátní zařízení. Proto byl zvolen výchozím modelem pro OpenCV. [5]

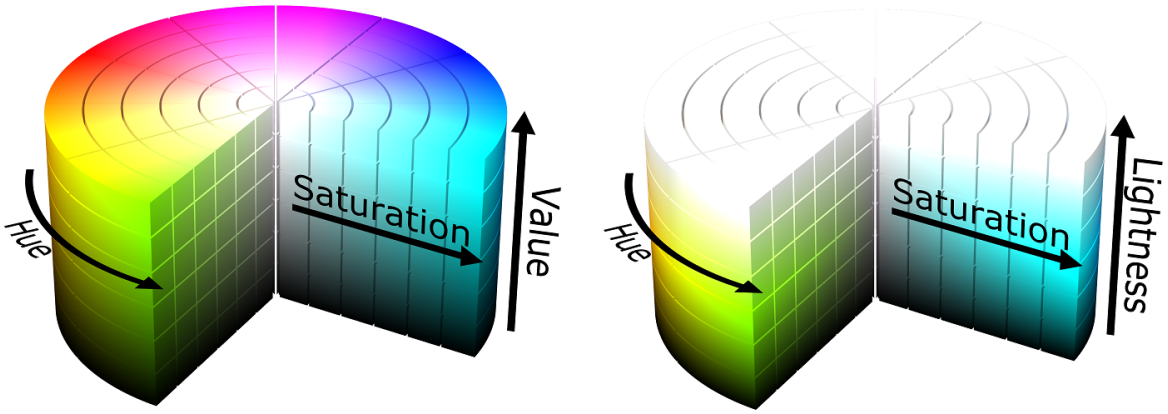
[1] Obrázky převzaty z Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation. Dostupné z: https://cs.wikipedia.org/wiki/RGB

3.2 Barevné modely HSV a HLS

Nejčastější model, který je využívaný pro detekci objektů v barevných obrazech, je HSV. Stejně jako u RGB se také jedná o aditivní model, který vytvořil v roce 1978 Američan Alvy Ray Smith.

HSV model lze reprezentovat pomocí válce, kde poloměr představuje sytost barvy (Saturation), se kterou docílíme určitého množství šedi v poměru k sytosti. Výška určuje hodnotu jasu, resp. množství bílého světla (Value). Tím se určí jak světlá nebo tmavá barva je a kolik světla lze odrazit. Tato hodnota je nahrazena v HSL modelu hodnotou svítivosti (Lightness). Úhlová hodnota (Hue) představuje barevný odstín, který se nastavuje pomocí 360° barevného kola. [6]

Grafické znázornění těchto modelů je na obr. č. 6.



Obrázek 6: Válcové znázornění modelů HSV a HLS[1]

Další společnou vlastností s RGB modely je jejich vyjádření v počítačových systémech. HSV a HLS se dají také vyjádřit jako osmice bitů pro každou svou složku. Nesmíme ale zapomenout, že maximální hodnota odstínu není 255, jak tomu bývá u RGB složek, ale pouze 180. Tato hodnota totiž reprezentuje nejvyšší možný konvexní úhel. Následné překročení této hodnoty může vést k chybným interpolacím.

Důvodem využívání HSV modelu v segmentování je, že se lépe určují limitní hodnoty pro možné prahování. Tuto vlastnost poskytuje přímo rozvržení modelu, díky kterému můžeme separovat barvu od intenzity (a svítivosti). Například pokud potřebujeme prahovat šedou až bílou barvu, tak je nutné nastavit limity v RGB u všech složek. U HSV modelu stačí pouze nastavit intenzitu barvy.

[1]Obrázky převzaty z Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation. Dostupné z: https://en.wikipedia.org/wiki/HSL_and_HSV

3.2.1 Převod RGB do HSV a HLS

I přesto, že modely HSV a HLS se liší pouze v jedné veličině, vzorce pro převod z RGB spektra se mírně liší. K převodu RGB na HSV se využívají vzorce[1]:

$$
\begin{aligned}
H \leftarrow & \begin{cases} 60(G-B)/(V-min(R,G,B)) & pokud\ V=R \\ 120+60(B-R)/(V-min(R,G,B)) & pokud\ V=G \\ 240+60(R-G)/(V-min(R,G,B)) & pokud\ V=B \end{cases} \\
S \leftarrow & \begin{cases} \frac{V-min(R,G,B)}{b} & pokud\ V \neq 0 \\ 0 & ostatní\ případy \end{cases} \\
V \leftarrow & max(R,G,B)
\end{aligned} \tag{1}
$$

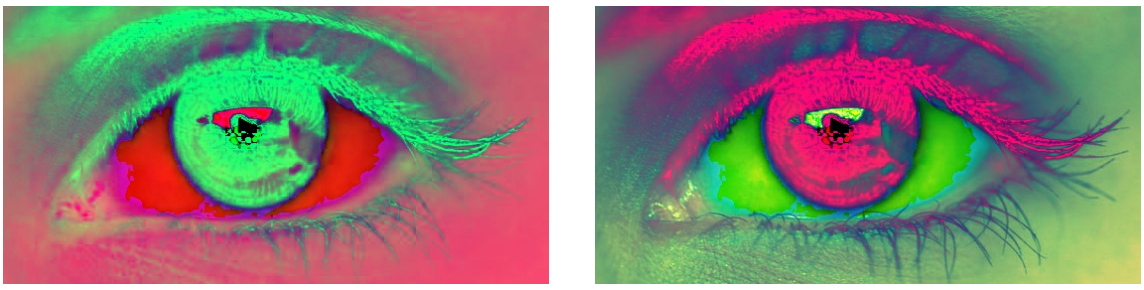
Zatímco k převodu mezi RGB a HLS se používají následující vzorce[1]:

$$
\begin{aligned}
H \leftarrow & \begin{cases} 60(G-B)/(V_{max}-V_{min}) & pokud\ V_{max}=R \\ 120+60(B-R)/(V_{max}-V_{min}) & pokud\ V_{max}=G \\ 240+60(R-G)/(V_{max}-V_{min}) & pokud\ V_{max}=B \end{cases} \\
L \leftarrow & \frac{V_{max}+V_{min}}{2} \\
S \leftarrow & \begin{cases} \frac{V_{max}-V_{min}}{V_{max}+V_{min}} & pokud\ L<0.5 \\ \frac{V_{max}-V_{min}}{2-(V_{max}+V_{min})} & pokud\ L \geq 0.5 \end{cases}
\end{aligned} \tag{2}
$$

kde

$$V_{max} = max(R,G,B) \qquad a \qquad V_{min} = min(R,G,B)$$

Po dosazení hodnot z RGB modelu, s ohlédnutím na hodnotu odstínu, můžou vzniknout výsledky uvedené na obr. 7.



(a) Oko ve modelu HSV

(b) Oko ve modelu HLS

Obrázek 7: Transformace oční části z RGB na HSV a HLS

[1]Vzorce převzaty z OpenCV: Open Source Computer Vision [online].
Dostupné z: http://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html

4 Operace určené k předzpracování a úpravě obrazu

Většina obrazů před a po libovolné segmentaci může obsahovat šum a další rušivé vlivy. Proto se na obrazy (nebo na jejich výřezy) používají operace, které pomáhají lépe izolovat detekované objekty. Ty mají za úkol eliminovat šum, rušivé vlivy a jiné drobné změny.

4.1 Morfologické transformace

Morfologické operace jsou metody, které se využívají pro odstranění šumu, zjednodušení tvaru, ale hlavně pro zdůraznění struktury (resp. izolování) objektů. Nejčastěji se s nimi setkáme při úpravě binárního obrazu, ale lze je použít i na barevné spektrum.

OpenCV nabízí celou škálu morfologických úprav, ale pro pouhé zvýraznění struktury se využívají metody: *eroze*, *dilatace*, *otevření*, *uzavření* a *morfologický gradient*.

Eroze je funkce, která odfiltrovává ztracené množiny bodů v obraze a tím odstraňuje drobné nerovnosti. Dochází v podstatě ke zmenšení obrazu, díky čemuž je vhodně využitelná pro izolování očního bělma. Dilatace je pravým opakem eroze. Ta se právě snaží zvětšit osamocené množiny bodů. Tím hledané objekty zvětšuje a malé díry v obraze nechává nabobtnat. Tato metoda je zase výhodná při detekci očního bělma pomocí prahování.

(a) Binární tvar oka před morfologií (b) Binární tvar oka po erozi (c) Binární tvar oka po dilataci

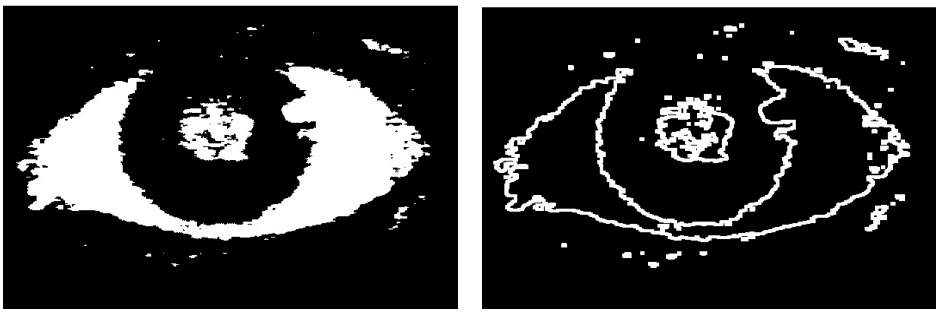
Obrázek 8: Příklad eroze a dilatace na binárním tvaru oka

Kromě eroze a dilatace se navíc využívají funkce *otevření* nebo *uzavření*. Jedná se o metody, které jsou pouhé kombinatoriky eroze a dilatace, ale jejich praktičnost zajišťuje vysokou míru použitelnosti.

(a) Originální binární obraz (b) Otevření (Dilatace po erozi) (c) Uzavření (Eroze po dilataci)

Obrázek 9: Příklad operací Opening a Closing na binárním tvaru oka

Poslední operací je *morfologický gradient*. Ten nám vytváří binární rozdíl mezi dilatováním a erozí na aplikovaném výřezu a je možné ho využít jako doplňkovou součást detektorů kontur.



(a) Před využitím operace morf. gradientu (b) Po využití operace morf. gradientu

Obrázek 10: Využití operace morfologický gradient na Otsu transformaci

4.2 Využití barevných map

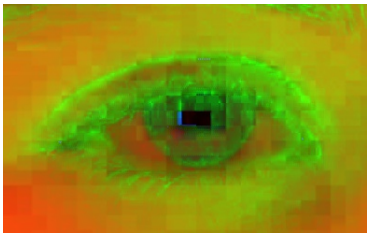
Využití barevných map je technika, která se může použít před aplikováním barevného prahování. Tato metoda se je vhodná tehdy, když potřebujeme snímek z odstínů šedi převést do barevných modelů. Barevná mapa překryje celé spektrum odstínu šedi a nahradí ho svým vlastním spektrem.

Obrázek 11: Příklad záměny odstínů šedi do barevné mapy HOT

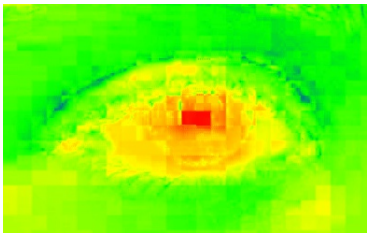
Při detekci očního bělma je možné tyto mapy využít, když chceme včetně skléry zobrazit i duhovku oka. Tím se také může mírné rozložit intenzita barev ve snímku a prahování je poté jednodušší. Například se nám eliminují oblíčky ve tvářích, nebo v případě detekce tváře z částečného profilu vymizí objekty umístěné za postavou.

Knihovna OpenCV nabízí dvanáct barevných map, které lze využít:

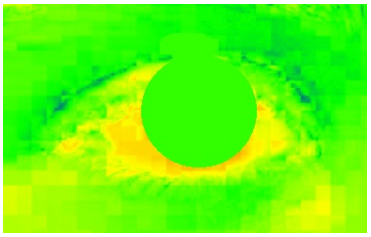
- Autmn
- Bone
- Hot
- HSV
- Jet
- Ocean
- Pink
- Rainbow
- Spring
- Summer
- Winter



(a) HSV oko před využitím mapy



(b) Aplikace mapy RAINBOW



(c) Průnik s vygenerovanou duhovkou

Obrázek 12: Využití barevné mapy na oční vrstvě

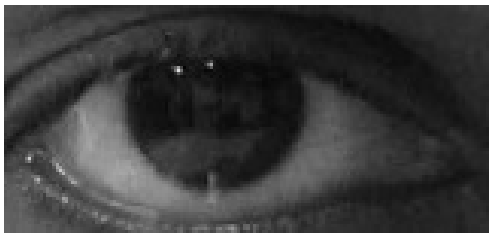
Ze snímků č. 12 lze zpozorovat sytě oranžovou barvu v levém dolním rohu před aplikováním barevné mapy. Ta je shodná s barvou představující oční bělmo v HSV. Proto, pokud již není možné snížit rozlišení sestrojeného obdélníku kolem oka, je nutné využít namapování.

V našem příkladě je po mapování žlutá, až žluto-oranžová barva součástí skléry, zatímco žluto-oranžová až červená barva je součástí duhovky. K eliminaci těchto buněk je možné využít barevné prahování, nebo detekci kružnic. Tuto kružnici můžeme vyhledat např. pomocí Houghovy transformace z kapitoly č. 5.1. Po průniku naprahované oblasti s vygenerovanou duhovkou nám, v ideálním případě, zbude pouze oční bělmo.

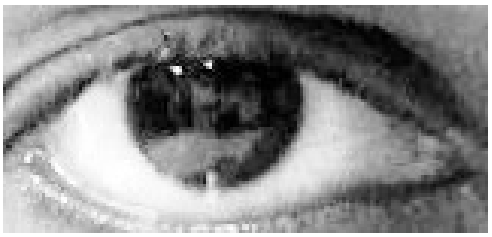
4.3 Rozmazávání a vyvažování obrazů

Dalšími technikami určené k předzpracování obrazu je rozmazávání (neboli rozostření) a vyvažování. Rozostření obrazu je efektivní metoda pro odstranění šumu z obrazu. OpenCV nabízí pro tuto funkci mnoho metod, mezi které patří Gausova metoda, filter2D, rozostření dle mediánu nebo metoda Smooth. Tyto technologie jsou nutné pro detekční metody, které vyžadují na svém vstupu rozmazaný obraz (např. Hough. transformace). [7]

Problematika vyvažování obrazu se používá tehdy, když je obraz v některých místech tmavší nebo světlejší než v ostatních částech. V OpenCV se využívá k vyvažování funkce *equalizeHist* (vyvažování histogramu), která tyto místa rozloží po celém spektru.



(a) Oko před vyvážením



(b) Oko po vyvážení histogramu

Obrázek 13: Oko před a po histogramovém vyvážení

5 Principy detekce duhovky pro následnou extrakci bělma

Mnoho technologií, zabývajících se sledováním lidské tváře, využívá detekci duhovky. Vyhledání bělma totiž může sloužit jako doplňková součást takových detektorů. Matematicky lze totiž tvrdit, že pokud správně nalezneme duhovku, tak se bělmo bude vyskytovat na minimálně jedné straně, resp. někde blízko souřadnice u $\pm 90°$ až $180°$ od nejvýše položeného bodu prstence duhovky. Záleží hlavně na směru pohledu. Stejně tak je možné nalézt korektně duhovku oka, pokud máme správně naprahované bělmo.

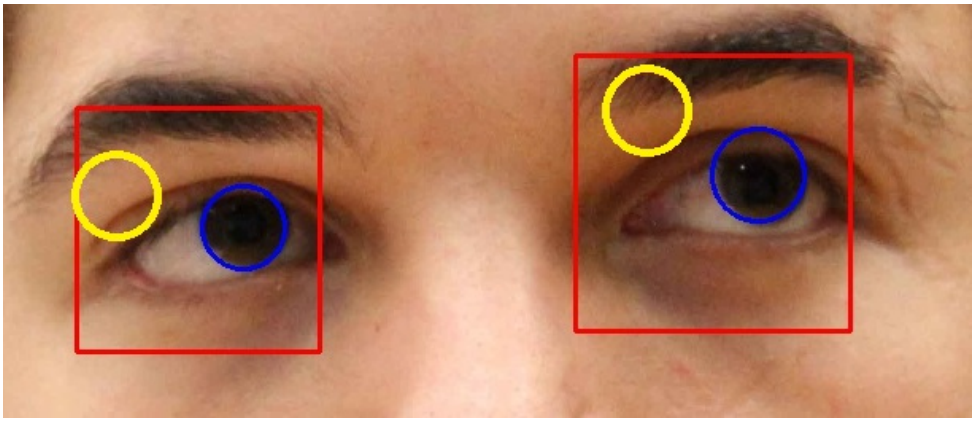
Získání pozice duhovky lze provést několika způsoby, ale v mnoha případech je nutné využít detektor kružnic. Také se ale můžeme setkat s detekcí duhovky vyhledávanou za pomocí Sobel derivace, nebo detekcí duhovky pomocí Otsu transformace a následnou analýzou hran. Vhodným způsobem je Houghova detekce kružnic.

5.1 Detekce duhovky pomocí Houghovy transformace

Houghova transformace se využívá k detekci elementárních objektů, jejichž hranice lze popsat jednoduchými křivkami. Např. v případě kružnice pomocí vzorce $(x-a)^2 + (y-b)^2 = r^2$. Výhodou tohoto typu transformace je robustnost vůči vstupním datům, které nemusí být ve všech případech příliš kvalitní. Transformace využívá gradienty, které obsahují informace o hranách a dále Cannyho detektor hran. [8]

Princip detekce kružnic spočívá v nalezení hodnot (a, b), které představující střed kružnice a veličiny r reprezentující poloměr. Pro každý bod z obvodu se mění souřadnice středu a poloměr se dopočítává. Pokud se nějaký střed pro konkrétní poloměr vyskytuje příliš často, tak je pravděpodobné, že se jedná o hledané souřadnice pro poloměr r. [9]

Pokud nastavíme ideální parametry pro velikost duhovky a minimální vzdálenost mezi středy detektovaných kružnic, algoritmus nalezne většinu možných prstenců, kde ale pouze jeden z nich pro každé oko představuje bělmo. Příkladem může být obr. 14

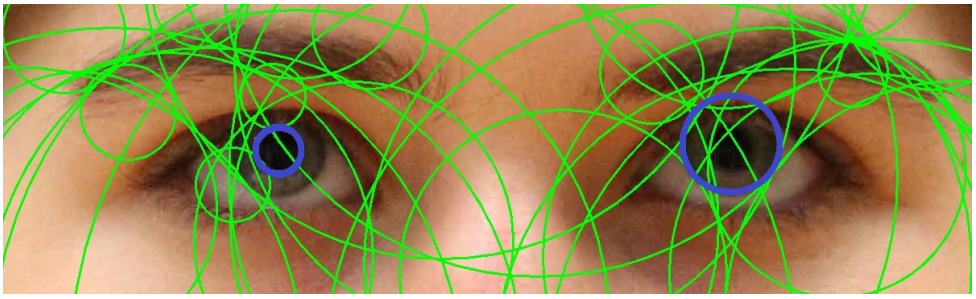


Obrázek 14: Příklad výstupu Houghovy detekce kružnic

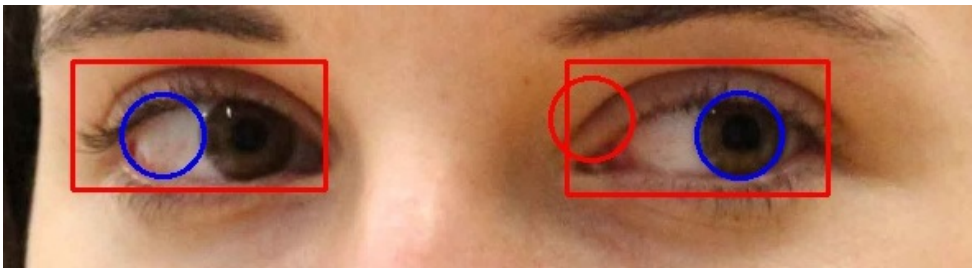
Z výstupu je patrné, že kružnice po obvodu zvýrazněné modrou barvou jsou duhovky, zatímco kružnice, které s duhovkou nemají nic společného, jsou označené žlutě. Tento výstup vznikl za pomocí výpočtu průměrné sytosti obsahu.

Uvnitř kružnic byl sestaven vepsaný čtyřúhelník, ve kterém se pomocí cyklů procházely jednotlivé buňky, a tím se provedl výpočet průměrné hodnoty sytosti. Po kalkulaci byly vyhledané kružnice seřazeny sestupně dle hustoty barvy a následně byla vybrána první kružnice pro každé oko. Tato nejtmavší kružnice je potencionální duhovkou.

Tento způsob ale závisí na správném nastavení parametrů detekce. Je doporučováno takové nastavení, aby detektor nalezl přiměřený počet kružnic. Pokud se nalezne pouze jedna, tak je možné, že místo duhovky byla nalezená nějaká možná součást očního víčka. Na druhou stranu pokud počet kružnic bude příliš vysoký, tak se může detekovat místo duhovky zornice. Příklady takových Houghových detekcí jsou na obr. č. 15



(a) Výstup Houghovy detelce při vysoké koncentraci kružnic



(b) Výstup Houghovy detekce při nízké koncentraci kružnic

Obrázek 15: Příklady výstupů Houghovy transformace s nízkou a vysokou koncentrací kružnic

Je také možné využít vyhledání kružnic v naprahovaném prostředí. Otsu transformace, popsaná v kapitole 6.4, nám zajišťuje, že nejtmavší oblasti výřezu oka, které většinou představují duhovky a oční víčka, jsou binárně odlišeny od ostatních naprahovaných objektů. Houghova transformace ale nelze využít na výstup Otsu prahování. Proto je nutné tento výstup převést zpět do černobílého formátu.

6 Jednotlivé přístupy řešení detekce bělma

V této kapitole jsou popsané metody, které lze experimentálně využít pro detekci bělma. Většina z popsaných metod využívá tzv. segmentaci, tj. určení oblastí v obraze, jež odpovídají hledaným prvkům.

Mezi tyto metody patří prahování v barevném modelu nebo v modelu v odstínech šedi. Další, sice méně častěji využívanou metodou, je adaptivní prahování s detekcí kontur. Jednou z automatických metod detekce je Otsu transformace (neboli binarizace). S touto metodou ale musíme využít jeden z vhodných algoritmů určených pro vyhledání limitních prostorů. Mezi takové metody patří například Watershed nebo FloodFill. Poslední popsanou metodou je detekce bělma z význačných bodů ve tváři.

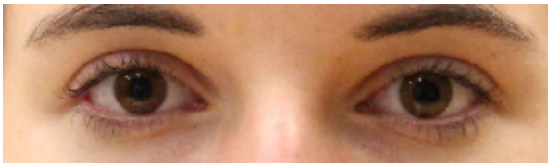
6.1 Prahování očního bělma pomocí v barevných modelech

Prahování v barevném spektru je jedna z nejjednodušších metod segmentace obrazu. Výstupem je obvykle binární obraz, který je reprezentován pomocí nul a jedniček. Díky své rychlosti se jedná o dobře využitelnou metodu pro detekci bělma, ale bohužel zde existují rušivé vlivy, jako například častá možnost záměny cizího objektu (nejčastěji tváří) za hledanou skléru. Proto je nutné tyto vlivy sledovat a pokusit se je eliminovat kalibrací tzv. prahovacích limitů.

Právě na jejich funkci je prahování v barevných modelech založené. Princip totiž spočívá ve dvou proměnných představující limity pro prahování. Pokud prvek v obraze (pixel) je usazen mezi stanovenými limity, změní se jeho hodnota na jedničku. V opačném případě se změní jeho hodnota na nulu.

Prahování bělma může probíhat následovně: Na vstupní pár očí (obr. 17a) se aplikují zadané odpovídající limity pro bělmo v daném modelu. Tím vznikne binární snímek, který obsahuje naprahované buňky (obr. 17b). Následně se pomocí morfologických operací mohou izolovat místa, které představují možnou část bělma. (obr. 16c). Někdy je ovšem tento krok ignorován. V posledním kroku se tyto bílé místa aplikují na vstupní snímek (obr. 16d).

Pokud využijeme metodu barevných map, tak je prahování mírně odlišné. Je totiž vhodné, aby došlo k překrytí výstupu prahování s detekovanou duhovkou. Tu obvykle do výstupu musíme ukotvit.



(a) Vstupní pár očí určených pro detekci

(b) Po aplikovaní prahování

(c) Po aplikování morfologických operací

(d) Výstupní pár očí se zvýrazněným bělmem

Obrázek 16: Příklad prahování na HSV modelu bez využití barevné mapy

(a) Prahování před vložením duhovky

(b) Prahování po integraci duhovky

Obrázek 17: Příklad prahování na HSV modelu s využitím barevné mapy

S metodou prahování v barevném prostředí pracovalo mnoho výzkumníků. Například P. R. Periketi, ve své práci *Gaze Estimation Using Sclera and Iris Extraction* porovnával metody jednoduchého a adaptivního prahování pro detekci pohledu, resp. účinnost detekcí duhovky a bělma. Sestavil tabulkové hodnoty pro RGB, HSV, CIE a další modely, které následně testoval. Sám ovšem odmítal čisté pracování v RGB modelu a proto ho mechanicky předpřipravil pro vyextrahování objektů. [10]

Další slibnou metodu segmentace bělma navrhl D. P. Mohapatra. Ten v knize *Intelligent Computing, Networking and Informatics* popsal vlastní postup pro jeho identifikaci. Nejdříve převedl RGB model do HSI[1] a následně nahradil saturační hodnotu. Poté vypočítal průměrnou intenzitu v okolí 17x17 pixelů okolo všech naprahovaných bodů a porovnával s černobílým obrazem.

[1]Model s podobnými vlastnostmi jako HSL, ale OpenCV nemá integrované vzorce pro jeho převod. Přesto je ale možné z teoretických vlastností intenzitu I vypočítat

6.2 Metoda prahování v černobílém prostředí

Existuje také možnost přímého prahování v černobílém prostředí. Obraz je v této metodě pouze převeden do černobílého formátu a následně je v něm naprahováno bělmo s využitím jedné limitní hodnoty. Pro lepší realizaci je také před prahováním aplikováno vyvážení histogramu. Metoda detekce s jedním prahem je ale využitelná spíše pro vyhledávání symbolů a ne práci s fotografiemi, kvůli vysoké míře šumu.[12]

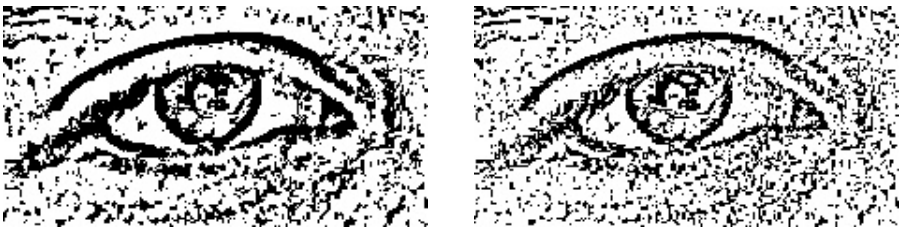
Obrázek 18: Jednoduché prahování v černobílém prostředí při limitní hodnotě = 100

Z výstupů prahování lze vidět, že oční bělmo by ještě na jedné straně šlo detekovat, zatímco na druhé už ne. Kvůli slabé segmentaci tudíž není tahle funkce doporučována, leda bychom využili detekci hran, nebo na výřez odstínu šedi aplikovali metodu MSER či FloodFill. Tyto funkce ale nejsou v této metodě příliš využívané.

6.3 Využití adaptivního prahování s detektorem kontur

Další možnou metodou vyhledávání bělma je využití detektorů kontur na výstup adaptivního prahování. Rozdíl mezi adaptivním a globálním prahováním je, že adaptivní rozkládá rovnoměrně světelnou intenzitu. Globální prahování je tedy aplikováno na celý snímek, zatímco u adaptivního závisí limit na nastavených parametrech.

OpenCV deklaruje dvě metody prahování tohoto typu: Mean a Gaussian. Tyto metody se liší počítáním průměru ve stanovených blocích. V Mean adaptivním prahování mají všechny pixely stejnou prioritu, zatímco v typu Gaussian jsou váhy pixelů určeny Gaussovou funkcí podle vzdálenosti od středu.



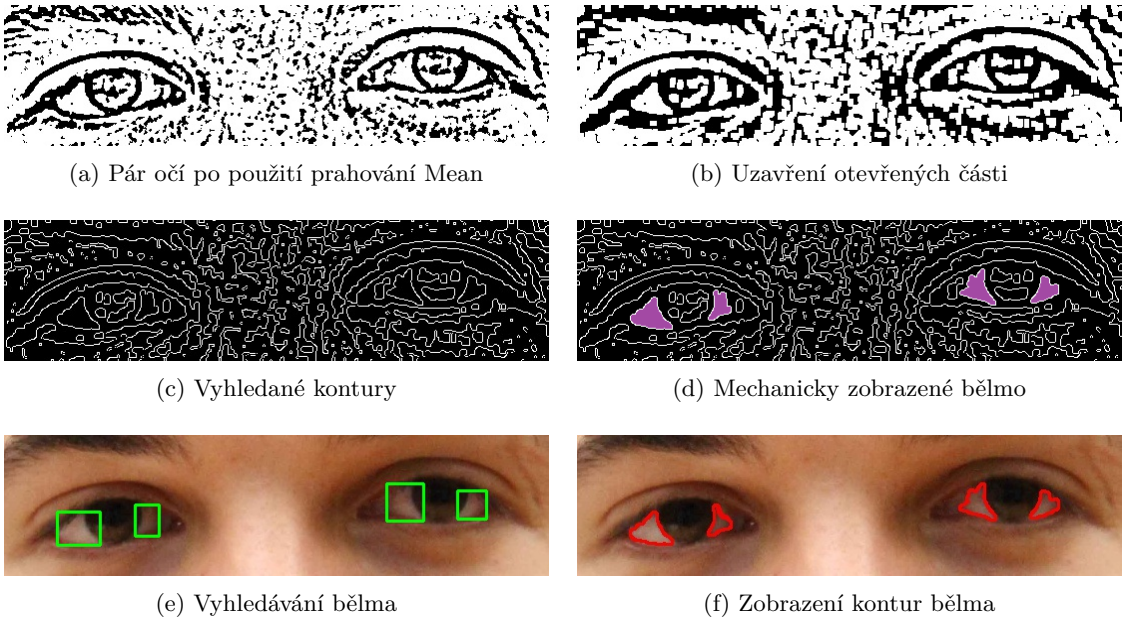
(a) Prahování Mean (b) Prahování Gaussian

Obrázek 19: Metody adaptivního prahování

Z těchto výstupů nyní lze detekovat polygony představující skléru. Můžeme využít Cannyho detektor, který slouží k detekování hran, nebo použít detektor kontur. Kontury jsou získávány právě z hran, ale navíc vyžadují, aby detekovaná kontura byla zároveň uzavřeným polygonem. Jelikož je bělmo, jakožto objekt, v obrazech vždy uzavřené a nikdy nepřetéká mimo svou vlastní pozici, tak je na místě detektor kontur využít.

Obr. 20d představuje bělmo, které je nutné vyhledat. Tato část byla vložena mechanicky (ručně do snímku). Jednou z vhodných možností, jak tuto část vyhledat je buď výpočet histogramu, nebo vypočítat průměrný odstín, jak tomu bylo v kapitole *Detekce duhovky pomocí Houghovy transformace* (č. 5.1). Také je možné použít algoritmus pro zjištění počtu bílých míst uvnitř kontury nebo prozkoumat okolní buňky. Možností je ale mnohem víc.

Kontura je nepravidelným geometrickým obrazcem. Proto ji není možné procházet buňku po buňce. Proto existuje v OpenCV metoda *boundingRect*, která okolo polygonu vykonstruuje obdélník, který již lze procházet pomocí cyklů. Následně lze např. vypočítat sytost kontury a následně kontury s vysokým obsahem bílé barvy zobrazit.



(a) Pár očí po použití prahování Mean

(b) Uzavření otevřených části

(c) Vyhledané kontury

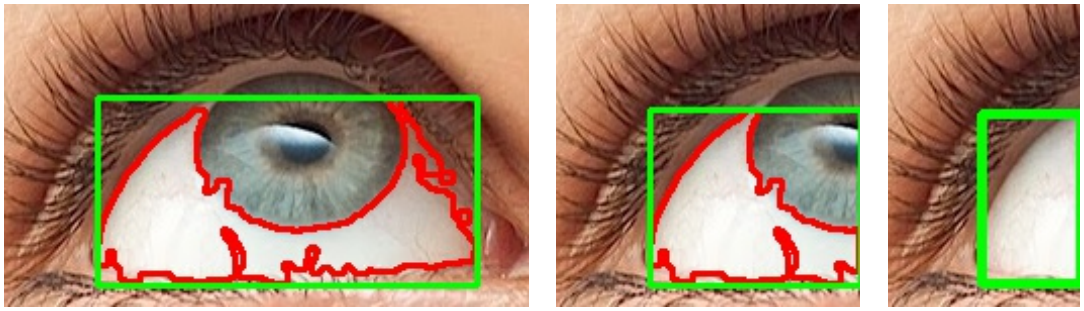
(d) Mechanicky zobrazené bělmo

(e) Vyhledávání bělma

(f) Zobrazení kontur bělma

Obrázek 20: Příklad prahování bělma pomocí detektoru kontur

Bohužel, systém detekce s využitím kontur nemá jednoduché řešení. Existují zde hrozby, jako například sledování objektu v horní části zorného pole. Například když se podíváme na obr. 21a, tak lze vidět, že opsaný obdélník nad konturou obsahuje i část duhovky, jejíž hustota barvy může výpočet sytosti velmi ovlivnit.



(a) Správná kontura v pohledu vzůru (b) Polovina kontury (c) Levá část

Obrázek 21: Příklad chybného prahování při sledování objektu mimo zorné pole

Obdélník zde nepřetéká pouze v malé části, ale v celé duhovce. Proto je zde nutné využít jiný algoritmus. I ty ale můžou selhat z důvodů vlivů prostředí, mezi které se například řadí osvětlení. Je možné, že světlo působící na lidskou tvář osvětlí jednu část, zatímco druhá zůstane ve stínech. Tyto vlivy ale můžou nastat ve všech segmentačních metodách. Nepomůže nám ani vyvážení histogramu, kde rovnoměrně rozložíme osvětlení. To nám pouze zajistí že oční skléra ztratí svou výjimečnou barvu.

Jednou z možností, jak se těmto problémům vyhnout, je odseknutí očního výřezu na části. Poté v těchto oblastech vyhledáme kontury. Zde je ale nutné ověřovat pozici duhovky tak, abychom nevyhledávaly kontury bělma v oblastech kde vůbec nejsou.

6.3.1 Využití algoritmu MSER

Algoritmus MSER (Maximally stable extremal regions) pracuje na principu postupného prahování oblasti. Pokud zůstanou neměnné přes definovaný počet prahů, jsou prohlášeny za *stabilní*. Kolem těch se následně vytvoří tzv. konvexní obal. Ten ovšem narozdíl od detekcí kontur lze překrýt dalšími obaly, což přináší velkou nevýhodu. Vzniknou tím totiž duplicity, u kterých musíme ověřovat, zda jsou stále součástí bělma. To realizuje podobné až stejné problémy, jako v předchozí kapitole.

Obrázek 22: Aplikování MSER algoritmu na výřez oka

6.4 Detekce bělma pomocí Otsu prahování

Otsu transformace je jedna z nejčastěji využívaných typů detekce na systému prahování. Jeho časté využití je z důvodu automatického vyhledávání prahu na základě výpočtu rozptylu. Proto je nutné aplikovat prahování pouze na detekovaný výřez oka, namísto celé tváře.

Pokud tak využijeme, zajistíme, že předpokládané nejtmavší oblasti (např. duhovka) budou binárně označené černou barvou, zatímco světlejší oblasti jsou označené bíle. Na druhou stranu, pokud využijeme prahování na celé tváři a zobrazíme si výřez oka, tak většina objektů už bude nedetekovatelná. (viz. obr. č. 23)

Pokud známe pozici duhovky, např. zjištěním z Houghovy transformace, tak můžeme porovnávat její okolí s výstupem transformace. Pokud okolí bude vyznačené bíle, jedná se o pravděpodobnou součást skléry. Nejlépe je vhodné zahájit vyhledávání v dolní půlce duhovky, protože oční víčko v přímém pohledu do kamery vždy kryje její horní část.

(a) Oko při využití Otsu transformace na celé tváři (b) Oko při využití Otsu trans. ve vlastním výřezu

Obrázek 23: Oko po Otsu transformaci při využití prahování na celé tváři a na výřezu z tváře

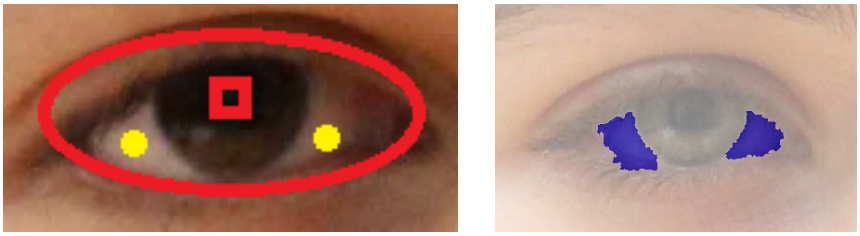
Následně musíme vyextrahovat okolní bělmo za pomocí algoritmů určených pro spojení podobných komponent. Mezi tyto algoritmy se řadí primárně tyto metody:

- Watershed algoritmus
- FloodFill algoritmus

6.4.1 Využití algoritmu Watershed

Watershed metoda je jedna z nejpoužívanějších segmentačních metod určených pro oddělení cizích objektů. Z anglického originálu lze přeložit jako metoda zaplavování. Pro využití této metody potřebujeme implementovat tzv. značkovací mapu, která je shodná s velikostí vstupního obrazu nebo výřezu. Do této mapy vkládáme markanty, neboli body v oblasti, kterou chceme zaplavit. Pro detekci bělma je vhodné prvním markantem označit centrální oblast duhovky oka a úplně stejným markantem okolí oka, které můžeme interpretovat širší elipsou okolo duhovky. Druhým markantem označíme oblasti, ve kterých předpokládáme existenci bělma. To můžou být například prostory okolo duhovky oka, jejichž hodnota v Otsu prahování je nenulová (bílé oblasti).

Po spuštění Watershed algoritmu se následně zaplavuje okolí nastavených značek (markantů) od jejich lokálních minim až po lokální maxima, dokud nebude celá značkovací mapa zaplněná. Jinými slovy se markanty budou rozšiřovat do té doby, dokud v mapě bude existovat nezaplněné místo. Následně vyextrahujeme oblasti, které jsou zahlcené druhým markantem, které bělmo představují. Na obr. 24a lze vidět využití popsaných markantů.



(a) Vhodné využití markantů pro identifikaci bělma

(b) Výstup Watershed algoritmu

Obrázek 24: Příklad využití Watershed algoritmu

Samozřejmě není nutné využívat Otsu prahování, ale v případě, že máme nakloněný obraz (např. otočenou hlavu), tak je možné že místo bělma bude detekována kůže. Tím se markant č. 2 v nejhorším možném případě rozšíří po celé mapě. Existují sice možnosti, jak ověřit, zda bod v obraze je součást kůže, ale tyto typy detekcí jsou mírně nespolehlivé, protože jejich aplikování musíme podmiňovat odstínem pleti, resp. skupinou odstínů pleti, které mohou být individuální pro každého jedince.

6.4.2 Využití algoritmu FloodFill

Algoritmus FloodFill, neboli semínkové zaplavování je druhá možnost segmentace na podobném způsobu, jako Watershed. Zde ale nevyužíváme markanty, ale tzv. semínka, která vkládáme do semínkové mapy.

Do této mapy (resp. zahrádky) se zasazují semínka tam, kde je možný výskyt bělma. FloodFill algoritmus prohledává okolní pixely a není-li u nich nalezena výrazná odlišnost, připojí je k dosud vytvořené oblasti.

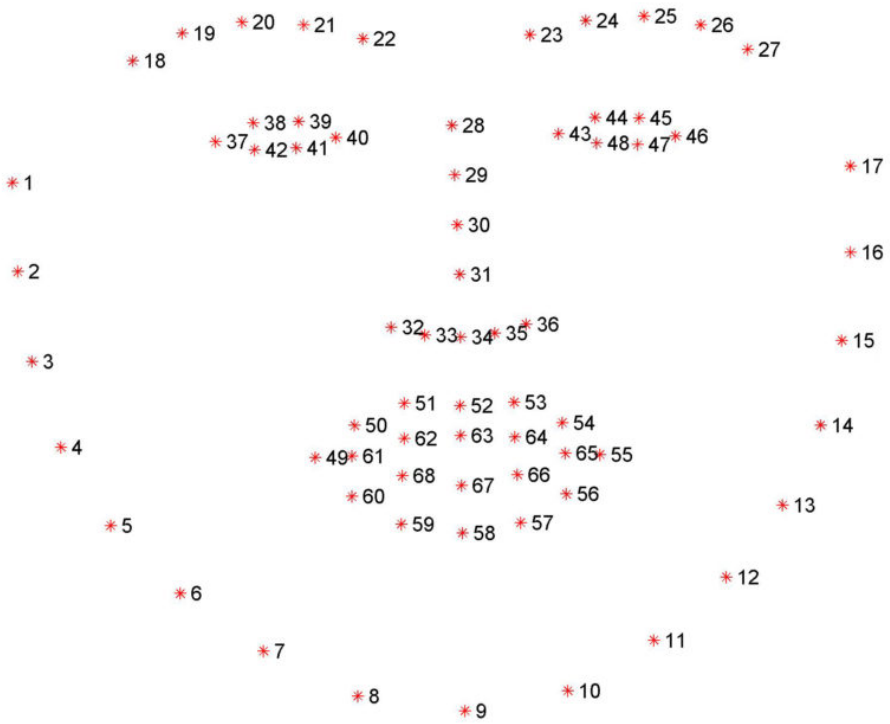
Obrázek 25: Příklad využití FloodFill algoritmu

Pokud opět využijeme znalost lokace duhovky, tak jsme schopni určit vhodné body, kde by se měla duhovka nacházet. Tyto body můžeme následně zaznamenat jako zasazená semínka. Algoritmus poté semínka rozšíří do vhodných oblastí. Výhodou (a možná i nevýhodou) je, že využíváme pouze jednu značku, zatímco v technice Watershed je jich nutné využít několik. Problém ovšem nastane, když vložíme semínko do bodu, které obsahuje nějakou vadu (např. přímo do kůže). V tomto případě se nemusí rozrůstat do správných prostor.

I u této metodiky můžeme ignorovat Otsu prahování, ale nastanou stejné problémy, jako v minulém řešení.

6.5 Detekce na bázi význačných obličejových bodů

Vyhledávání bělma na bázi význačných obličejových bodů je časově nejnáročnější detekcí. Tento typ extrakce využívá predikci tvarů, které bývají často natrénované na mohutném korpusu podobných objektů. OpenCV dovoluje využívat knihovnu *dlib*, který obsahuje prediktor, jež byl implementován V. Khazemim a J. Sullivanem v tézi *One Millisecond Face Alignment with an Ensemble of Regression Trees*. Ten odhaduje celkově 68 bodů, určených ke strukturalizaci lidské tváře (obr.č. 26). Tyto anotace jsou součásti iBUG 300-W datasetu, na kterém byla *dlib* predikce natrénována.



Obrázek 26: Vygenerované obličejové body v lidské tváři[1]

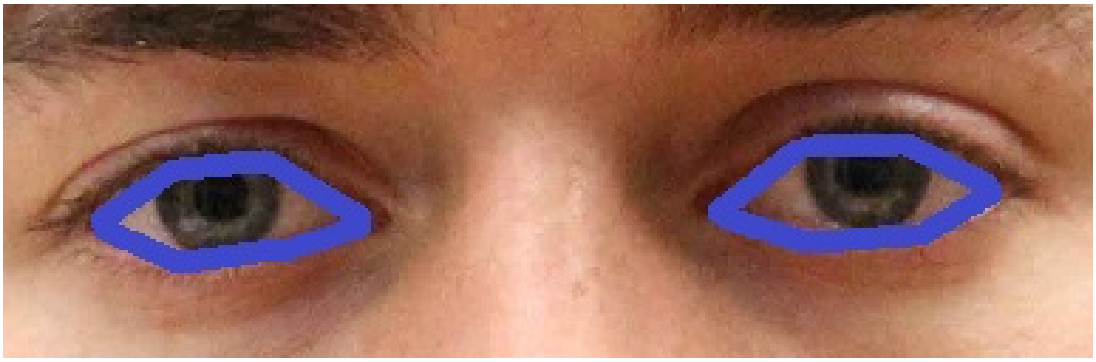
[1]Schéma polohovacích bodů byly převzaty ze článku *Facial landmarks with dlib, OpenCV, and Python* od Adriana Rosebrocka

Pokud se správně nashromáždí body uvnitř tváře, což bývá často podmiňováno kvalitním snímkem, bude levé oko v rozmezí 37-42 bodů, zatímco pravé oko bude tvořeno body 43-48. Z těchto bodů následně můžeme extrahovat bělmo.

Při vytvoření polygonu nad levým a pravým oko získáme výřez oční části. Tato část obsahuje jak oční bělmo, tak i duhovku. Proto, pokud vytvoříme nad těmito polygony opsaný obdélník a využijeme Otsu prahování, získáme duhovku obrazu. Jakmile tuto část vložíme zpětně do očních polygonů, zůstane pouze hledané bělmo.

Pomocí predikce význačných bodů můžeme celkem snadno analyzovat většinu problematik definovaných v úvodu této práce. Například se dá celkem snadno vypočítáním vzdálenosti mezi jednotlivými body duhovky ověřit, zda je oko otevřené, nebo pomocí sledování této veličiny počítat mrkání.

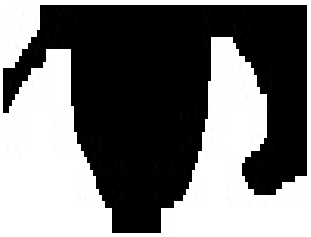
Nevýhodou této techniky je, že výpočty prováděné nad knihovnou *dlib*, trvají v řádu několika sekund. V případě využití detekčních snímků z knihovny CGDS, lze bez snížení rozlišení dosáhnout doby výpočtu až několika desítek sekund (Rozlišení snímků se pohybuje totiž kolem 18 Mpx). I přes tyto nevýhody lze s tímto řešením dosáhnout výborných výsledků detekce očního bělma.



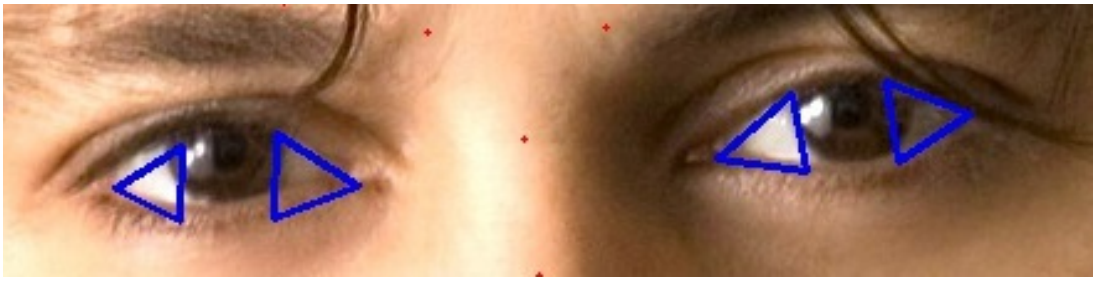
(a) Sestrojený polgyon nad očními body

(b) Vyplnění polygonu

(c) Průnik binární části OT s výplní polygonu



(d) OT výřezu



(e) Nevhodná metoda, kdy zobrazíme pouze boční části oši

Obrázek 27: Příklad využití význačných bodů při detekci bělma

7 Implementace a popis testovacích aplikací

V této předzávěrečné kapitole jsou uvedené postupy při řešení praktické části bakalářské práce, včetně využitých nástrojů. Jsou zde také popsány implementované metody detekce oční skléry a postup jejich průběhu včetně ovládání. Následně jsou využité technologie otestovány na volně přístupných vzorech CGDS. [13]

Vývojové prostředí

Detektor je vyvinut v programovacím jazyce Python. Tento interpretovaný jazyk poskytuje vhodné prostředí pro práci s algoritmy určené pro detekci objektů v obraze z knihovny OpenCV. Také obsahuje rychle aplikovatelné metody určené pro editaci vstupních snímků a podporuje práci s knihovnou *dlib*.

Pro vývoj detektoru očního bělma bylo potřeba nainstalovat následující aplikace a rozšíření:

- *Python Programming Language v.2.7.13* — Pro celkovou implementaci detektoru
- *Python OpenCv x86 library from OpenCv 3.2.0* — Pro přístup k detekčním alogitmům
- *Microsoft Visual C++ 9.0 for Python 2.7* — Pro instalaci zásuvných modelů
- *Numeric Python (numpy) v.1.12.0* — Pro obsluhu knihovny OpenCV
- *dlib for Python (dlib) v.18.17.100* — Pro možnost detekce s orient. body
- *PyBoost for Python (boost-python) v.1.63* — Pro správnou funkci knihovny *dlib*
- *ImUtils for Python (imutils) v.0.4.2* — Pro editaci vstupních snímků

7.1 Navržený algoritmus pro detekci očního bělma pomocí barevného prahování

K ovládání detekce očního bělma pomocí barevného prahování slouží sestrojený program *ColorThresholding.py*, který využívá metody z *knihovny ScleraDetector.py*. Ten se spouští z příkazové řádky s následujícími parametry:

```
usage: ColorThresholding.py [-h] [-image IMAGE | -images IMAGES]
[-color RGB,HSV,HLS,XYZ,LAB,LUV,YCrCb]
[-map AUTUMN,BONE,COOL,HOT,HSV,JET,OCEAN,
PINK,RAINBOW,SPRING,SUMMER,WINTER][-showIris]
```

IMAGE zde představuje jeden detekovaný snímek, zatímco IMAGES je celá množina jpg a png snímků v adresáři. Následně je také nutné pomocí parametru -color vybrat barevný model a je také možné využít barevnou mapu přes parametr -map. Můžeme pomocí parametrů určit, zda chceme i do výstupního souboru vložit detekovanou duhovku.

Program se ukončuje uzavřením uživatelského rozhraní *Program Settings*, nebo stisknutím klávesy ESC v okně *Sclera Detector*. Za předpokladu využití parametru -images je přechod na další snímek realizován klávesou SPACE.

Ideálně by implementovaný algoritmus pro prahování měl pracovat v následujících iteracích:

1. Převod vstupního snímku do odstínů šedi a do rozostřeného odstínu šedi
2. Detekce největšího objektu tváře ze snímku stupně šedi
3. Detekce dvou největších objektů očí, z levé a pravé horní poloviny detekované tváře, ze snímku stupně šedi

   (Výřez tváře byl rozdělen dvakrát na polovinu a oči se hledali v horních polygonech.)
4. Detekce duhovek pomocí HT v objektech očí z rozostřeného snímku stupně šedi

   (Duhovky byly zpracovány metodou vyjmutí nejsytějšího prstence, popsaného v kap.5.1)
5. Převod snímku do vybraného barevného prostředí (a vybrané barevné mapy)
6. Aplikování limitů prahování (z níže uvedené tabulky) pomocí uživatelského rozhraní na vybrané barevné prostředí
7. V případě využití barevných map vložení obsahu duhovek do výstupu prahování
8. Vložení výstupu prahování do vstupního snímku

Tabulka 1: Vhodné limitní hodnoty pro prahování bělma

	Minimální limitní hodnota	Maximální limitní hodnota
HSV	(0,0,110)	(140,110,255)
HLS	(0,100,0)	(180,255,70)
CIE	(90,0,90)	(255,130,255)
CIE L*a*b*	(110,1,1)	(255,135,255)
CIE L*u*v*	(110,0,0)	(255,113,255)

7.2 Navržený algoritmus pro automatickou detekci bělma pomocí Otsu transformace s využitím algoritmů Watershed a FloodFill

Druhým navrženým algoritmem je Otsu prahování s využitím Watershed a FloodFill. Ten se ovládá přes program *AutoSegmentation.py* opět přes příkazovou řádku s následujícími parametry.

usage: AutoSegmentation.py [-h] [-image IMAGE | -images IMAGES]
[-type Watershed,FloodFill] [-forceApply]
[-showIris] [-manualSegmentation]

Ovládání přechodu je stejné jako v minulé metodě, ale oproti barevnému barevnému prahování tento program vyžaduje parametr -type, který určuje vybraný algoritmus pro automatickou segmentaci. Následně je možno využít parametr -forceApply, který slouží pro ignorování Otsu transformace a přímé vkládání markantů nebo semínek kolem vygenerované duhovky. Využití parametru -manualSegmentation, který slouží pro ruční nastavení je popsáno níže.

Program pracuje obdobně, jako detektor na bázi barevného prahování. Využívá ale Otsu transformaci pro zvýraznění duhovky a jeho okolí. Následně se pomocí souřadnic detekovaných kružnic zjišťují hodnoty vyplynuté z prahování a tyto oblasti se testují kvůli prezenci bělma.

V případě algoritmu Watershed využíváme dva markanty. Jeden, který se sestrojí vně detekované kružnice a kolem duhovky s přihlédnutím na hodnotu šířky oka. Druhý se vytváří kolem duhovky na následujících souřadnicích, které reprezentují blízké okolí kružnic:

- $(x + r + 0.125r, y + r/2)$ za předpokladu, že hodnota v OT je nenulová
- $(x + 0.25r, y)$ za předpokladu, že markant nebyl vložen v předchozím bodě a hodnota prahování je nenulová
- $(x + r + 0.125r, y - r/2)$ za předpokladu, že markant nebyl vložen v předchozím bodě a hodnota prahování je nenulová

a na:

- $(x - (r + 0.125r), y + r/2)$ za předpokladu, že hodnota v OT je nenulová
- $(x - 0.25r, y)$ za předpokladu, že markant nebyl vložen v předchozím bodě a hodnota prahování je nenulová
- $(x - (r + 0.125r), y - r/2)$ za předpokladu, že markant nebyl vložen v předchozím bodě a hodnota prahování je nenulová

V případě algoritmu FloodFill se využívají pouze znalosti o kružnici (duhovky). Proto stačí vkládat semínka pro následný semínkový rozptyl do výše určených souřadnic.

Manuální nastavení markantů a semínek se ovládá pomocí počítačové myši (nebo touchpadu). Tato metoda slouží pouze k náhledu, možné detekce. U metody Watershed se v případě kliknutí pravého tlačítka vkládá markant číslo 1, zatímco v případě levého markant č. 2. U metody FloodFill se semínko nastavuje pomocí levého tlačítka myši. V případě kliknutí klávesy *R* se markantová, nebo semínková mapa vyčistí.

7.3 Algoritmus pro detekci bělma pomocí význačných bodů v obličeji

Algoritmus pro detekci bělma, který využívá význačné obličejové body je využíván přes program *FacialLandmarks.py* a vyžaduje také detektor *ShapePredictor.dat*.

usage: FacialLandmarks.py [-h] [-image IMAGE | -images IMAGES]

Tento algoritmus predikuje 68 bodů v obličeji, jak bylo popsáno v kapitole *Detekce na bázi význačných obličejových bodů* (č. 6.5) My ovšem potřebuje pouze body, které definují levé a pravé oko. Nad těmito body jsou vyplněné polygony (pomocí cv2.fillPoly) a sestrojen opsaný obdélník (cv2.boundingRect). Následně využijeme Otsu transformaci a s jejím výstupem provedeme průnik nad vyplněnými polygony. Postup je znázorněn na obr. 27

8 Testování aplikace

Veškerá funkčnost navržených algoritmů byla testována na datasetu CGDS. [13] Ten obsahuje přes 5880 snímků 56 lidí. CGDS byla v rámci této práce zmenšena, protože většina snímku přesahovala rozlišení 18 Mpx. Navíc byla v aplikaci využita knihovna *imutils*, určená pro změnu rozlišení. Tím se sníží výpočetní doba funkcí.

Na náhodně vybraných sadách, ze kterých bylo vybráno 100 snímků, bylo testováno využití modelu HSV, které se pro prahování v barvách nejčastěji využívá. Oční oblasti byly detekované s úspěšností 98%. Tento výsledek není překvapující, díky kvalitě testovaného datasetu. Průměrná rychlost detekce činila 42 milisekund s celkovou úspěšností 84%. Bělmo se v této metodě často rozlévalo mimo své hranice ze 32%. Po využití barevných masek se rozlévání snížilo na 11%, ale úspěšnost metody se také snížila na 62%. To bylo způsobeno nutností využití Houghova detektoru kružnic, který vkládal bělmo do prahování.

	#1	**#2**	**#3**	**#4**	**#5**	**Celková úspěšnost**
Detekce bělma	82%	85%	84%	87%	82%	84%
Významné opuštění prostoru	28%	34%	32%	35%	30%	32%

Tabulka 2: Testování detektoru při barevném prahování v modelu HSV

	#1	**#2**	**#3**	**#4**	**#5**	**Celková úspěšnost**
Detekce bělma	62%	57%	61%	63%	65%	62%
Významné opuštění prostoru	8%	12%	10%	12%	14%	11%

Tabulka 3: Testování detektoru při barevném prahování v modelu HSV s využitím masek

Další testovanou metodou bylo automatické nalezení prahu a využití algoritmů Watershed a FloodFill. Bohužel, aplikování těchto funkcí je podmíněné správnou detekcí oční duhovky. Přesnost nalezení duhovky se pohybovala kolem 78%. Rychlost detekce byla o něco vyšší než při limitování barev. Průměrná rychlost algoritmu Watershed činila 92 milisekund. Naopak rychlost FloodFill byla o poznání zvýšená, kolem hodnot 47 milisekund.

	#1	#2	#3	#4	#5	Celková úspěšnost
Detekce bělma	86%	82%	80%	84%	82%	83%
Významné opuštění prostoru	18%	12%	14%	13%	10%	13%

Tabulka 4: Detekce bělma pomocí algoritmu Watershed po Otsu transformaci

	#1	#2	#3	#4	#5	Celková úspěšnost
Detekce bělma	72%	80%	80%	72%	76%	75%
Významné opuštění prostoru	10%	16%	20%	16%	18%	16%

Tabulka 5: Detekce bělma pomocí algoritmu FloodFill po Otsu transformaci

Poslední testovanou metodou byla detekce význačných bodů v obličeji. U této časově náročné detekce není možnost významného opuštění hranic bělma. I přes svou nízkou rychlost, která se pohybovala kolem 3.5 sekund, je tato detekce nejúspěšnější.

	#1	#2	#3	#4	#5	Celková úspěšnost
Významné opuštění prostoru	88%	95%	92%	93%	84%	90%

Tabulka 6: Detekce bělma pomocí význačných bodů v obličeji

9 Závěr

Cílem bakalářské práce bylo porovnání metod, určených pro detekci očního bělma v obrazeh. V teoretické části byly popsána knihovna OpenCV a metody, které se využívají při detekci předem určených objektů v obrazech. Dále byly popsány jednotlivé barevné modely, které detektory často využívají. Také byly zdokumentovány metody pro předzpracování obrazu, princip vyhledávání duhovky a jednotlivé realizovatelné detektory očního bělma, mezi které patří i detektor kontur.

V rámci realizace praktické části jsem narazil na problémy z oblasti detekce duhovky. Právě ta se ukázala jako největší problém v následném prozkoumávání jejího okolí pro následnou detekci bělma z transformovaných oblastí.

V praktické části byl realizován poloautomatický a automatický detektor s využitím knihovny OpenCV, který buď prahoval bělmo v barevných modelech, nebo se vyhledávalo bělmo automaticky v okolí duhovky. Také byl navržen systém pro detekci bělma pomocí význačných bodů v obličeji, který se ukázal jako nejúspěšnější realizovanou detekcí v této tézi. Nejméně úspěšnou detekcí bylo využití barevných masek, do kterých musely být vloženy duhovky očí. Právě detekce kružnic byla nejméně úspěšná.

Téma bakalářské práce bylo pro mne přínosná, protože jsem získal znalosti z technologie určené pro zpracování obrazu. Tato práce se dá dále rozšiřovat, například stálým sledováním lidské tváře v případě nezachycení objektu v nové iteraci, nebo využitím neuronových sítí, určených pro detekci očního bělma.

Literatura

[1] PULLI, Kari. BAKSHEEV, Anatoly. KORNYAKOV, Kirill. ERUHIMOV, Victor. *Realtime Computer Vision with OpenCV* [*online*]. Jun. 2012, vol. 55, iss. 6. Dostupné z: `https://research.nvidia.com/`. ISSN 0001-0782.

[2] MERTA, M. *Detekce obličejů pomocí příznakového rozpoznávání.* Ostrava, 2012, 41 s. Bakalářská práce na fakultě informatiky Vysoké školy Báňské. Vedoucí práce: Ing. Fusek Radovan, Ph.D

[3] VIOLA, P - JONES, N. *Rapid Object Detection using a Boosted Cascade of Simple Features*: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 2001, vol.1, ISSN 1063-6919

[4] KOHOUT, V. *Historie a elementární základy teorie barev* Školská fyzika, roč. 2013, č. 1, s. 35-4255.

[5] NAYAK, S. *Why does OpenCV use BGR color format?* URL: `https://www.learnopencv.com/why-does-opencv-use-bgr-color-format/` [*cit.*$2015-27-09$]

[6] KERR. D. *The HSV and HSL Color Models and the Infamous Hexcones* [*online*]. Dostupné z:: `http://dougkerr.net/Pumpkin/articles/HSL_HSV.pdf` [*cit.*$2008-05-12$].

[7] VALENTA, M. *Porovnání výkonnosti počítání konvoluce v knihovnách I3dlibs a OpenCV.* Brno, 2015. 56 s. Bakalářská práce na fakultě informatiky Masarykovay univerzity. Vedoucí práce RNDr. Ulman Vladimír, Ph.D

[8] YUEN, H. K. – PRINCEN, J. - ILLINGWORTH, J., KITTER, J. *A comparative study of Hough Transform methods for Circle finding.* Newton, MA, USA: Butterworth-Heinemann, 1990, vol. 8, s. 71-77, ISSN 0262-8856

[9] Hledání parametrického popisu objektů pomocí Houghovy transformace [*online*] Dostupné z: `http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm`

[10] PERIKETI, P.R. *Gaze Estimation Using Sclera and Iris Extraction* Master Thesis on University of Kentucky, Electrical Engineering and Comp. Science dep. 2011, 41 s. Vedoucí práce Dr. Sen Ching Samson Cheung

[11] MOHAPARTA, D. P. *Intelligent Computing, Networking and Informatics.* Springer Publishing Company, Incorporate, 2014, 1314 s. ISBN 81-3221-664-4. Kapitola 2.1., Sclera Detection and Noise Removal, s. 1189-1192

[12] NIBLACK, W. *An introduction to digital image processing* Strandberg Publishing Company Birkeroed, Denmark, 215 s. ISBN 87-872-0055-4

[13] SMITH, B.A - YIN, Q. - FEINER, S.K. - NAYAR S.K.
Gaze Locking: Passive Eye Contact Detection for Human–Object Interaction.
Proceedings of the 26th ACM Symposium on User Interface Software and Technology, St. Andrews, Scotland, United Kingdom, 2013, s. 271-280. ISBN 978-1-4503-2268-3

[14] ROSEBROCK, Adrian. *Facial landmarks with dlib, OpenCV, and Python* [online]. [*cit.*2017 – 04 – 03]. Dostupné z: `http://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`

Seznam příloh

Příloha A: Struktura přiloženého optického média (1 strana)
Příloha B: Použité knihovny třetích stran (1 strana)

A Struktura přiloženého optického média

Adresář	Obsah
\src	Adresář s implentovanými aplikacemi
\cgds	Adresář s datasetem CGDS
bp.pdf	PDF soubor s textem bakalářské práce
mandatory.txt	TXT soubor s požadovanými knihovnami pro spuštění aplikace

B Použité knihovny třetích stran

- OpenCV - `https://sourceforge.net/projects/opencvlibrary/`
- dlib - `https://pypi.python.org/pypi/dlib/18.17.100`
- numpy - `https://pypi.python.org/pypi/numpy/1.12.1`
- imutils - `https://github.com/jrosebr1/imutils`
- boost-python - `http://www.lfd.uci.edu/~gohlke/pythonlibs/`